AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Previously Presented) A method for creating an activity within a process management system, comprising:

receiving first data reflecting a class file that implements an interface;

receiving second data reflecting a data representation file that specifies the environment, input parameters, and output parameters for the class;

packaging the first and second data; and

associating the packaged data with an activity that may be used in an automated workflow process to access information external to the process management system.

2. (Original) The method of claim 1, wherein the data representation file includes a section that determines the appearance of a representation reflecting the activity.

3. (Original) The method of claim 1, wherein the class file includes a method that is configured to obtain a value of a parameter defined in the data representation file.

4. (Original) The method of claim 1, wherein receiving first data reflecting a class file includes:

receiving data that defines a package for the class file; and

receiving data that defines methods that retrieve and set values to variables to be used by the activity.

5. (Original) The method of claim 4, wherein receiving data that defines methods includes:

receiving data that reflects a method that defines variables that are constant across all instances of the activity.

6. (Original) The method of claim 5, wherein the method is associated with an input hashtable to define values of a variable used by the activity

7. (Original) The method of claim 4, wherein receiving data that defines methods includes:

receiving data reflecting a method that defines values for variables in a first hashtable and retrieves values for variables from a second hashtable.

8. (Original) The method of claim 4, wherein receiving data that defines methods includes:

receiving data reflecting a method that releases resources used by an application that implements the activity when the application is unloaded from the process management system.

9. (Original) The method of claim 1, wherein receiving second data reflecting a data representation file includes:

receiving data reflecting a first section that defines a type and name of the class file;

receiving data reflecting a second section that defines parameters with values that remain constant within all instances of the activity;

receiving data reflecting a third section that sets values for selected parameters within a first hashtable;

receiving data reflecting a fourth section that defines what to do with parameters included in a second hashtable; and

receiving data reflecting a fifth section associated with a visual representation associated with the activity.

10. (Original) The method of claim 1, wherein packaging the first and second data includes:

packaging the first and second data into one of a JAR file or a ZIP file.

11. (Original) The method of claim 1, wherein associating the packaged data with an activity includes:

locating the packaged data; and

receiving data reflecting a visual representation that corresponds to the packaged files.

12. (Previously Presented) A method for implementing a custom activity within a process management environment, comprising:

defining a file associated with a custom activity; assigning a visual representation associated with the custom activity;

receiving an indication reflecting implementation of the custom activity in a workflow process based on a position of the visual representation in a process map representing the workflow process; and

invoking the file,

wherein the file includes a class file that implements an interface and a data representation file that specifies the environment, input parameters, and output parameters for the class.

13. (Original) The method of claim 12, wherein the file is an archive file an includes the visual representation.

14. (Currently Amended) A method for creating and defining a custom activity within a process management system, comprising:

creating at least one file defining properties associated with the custom activity; and

defining a model associated with the custom activity, wherein the custom activity may be used to access information external to the process manager system[[.]],

wherein the ~~a first~~ file includes ~~is~~ a class file that implements an interface and ~~a second file is~~ a data representation file that specifies the environment, input parameters, and output parameters for the class.

15. (Original) The method of claim 14, wherein the model is an image reflecting the custom activity.

16. (Original) The method of claim 14, further comprising: packaging the file and model into an archive file.

17. (Original) The method of claim 14, further comprising:

associating the custom activity with a workflow process managed by the process management system.

18. (Original) The method of claim 17, wherein associating the custom activity includes:

determining a position of the model in a visual process map reflecting the workflow process; and

invoking the custom activity in the workflow process based on the determination.

19. (Original) The method of claim 14, where in the at least one file includes a Java class file and an XML description file.

20. (Withdrawn) A method for implementing a custom activity in a process management system, comprising:

creating a process map reflecting an automated workflow process;

creating an image reflecting a custom activity; and

invoking a class defining the custom activity based on a manipulation of the image by a user such that the image is placed in the process map,

wherein the custom activity exchanges data with resources external to the process management system,

wherein at least two applications employ the custom activity, each class of the at least two applications is deployed to an appropriate folder in a class path such that the class path is shared by the at least two applications, and subsequent deployments of the at least two applications overwrite the previous application class that was deployed to the class path.

21. (Currently Amended) A method for creating a custom activity in a process management system, the custom activity exchanging information with resources external to the process management system, comprising:

receiving a first file reflecting a class file that implements an interface and a second file; and

archiving the files in an archive file such that when the custom activity is activated the archived files are accessed and executed,

wherein the first file and second file have the same root name and, once the archive file is created, a directory structure is checked to ensure that the directory structure reflects the package structure of class file.

22. (Original) The method of claim 21, wherein receiving the first and second files includes:

receiving package information associated with the first file that implement packages external to the process management system.

23. (Original) The method of claim 21, wherein receiving the first and second files includes:

receiving data that interacts with parameters associated with a hashtable defined in the second file.

24. (Original) The method of claim 21, wherein receiving the class and XML files includes:

receiving data associated with the second file that defines at least one hashtable used by the first file.

25. (Original) The method of claim 21, wherein the first file reflects a class file and the second file reflects an XML file.

26. (Original) The method of claim 21, wherein archiving the files includes:

archiving the files in an archive file consisting of one of a JAR file and a ZIP file.

27. (Previously Presented) A memory for storing data for access by a process being executed by a processor, the memory comprising:

a structure defining a class file that implements an interface and a data representation file that specifies the environment, input parameters, and output

parameters for the class, packaging the files, assigning an icon representing the packaged files, and associating the icon with an activity that performs processes defined by the class and data representation files.

28. (Original) The memory of claim 27, wherein the data representation file is an XML description file.

29. (Original) The memory of claim 28, wherein the XML description file defines the format of the activity.

30. (Previously Presented) A memory for storing data for access by a process being executed by a processor, the memory comprising:

a structure for maintaining an identity of a custom activity associated with a class file and a data representation file, parameters associated with the custom activity, a first hashtable reflecting data values to be used as input argument in a method, and a second hashtable reflecting output arguments of the method,

wherein the data of the first hashtable and arguments of the second hashtable are consistent with one or more data types specified within the data representation file.

31. (Previously Presented) A memory for storing data for access by a process being executed by a processor, the memory comprising:

a structure for defining a value of a parameter associated with an input hashtable, mapping a value of a parameter associated with an output hashtable, and

defining a user interface associated with a custom activity, associated with a class file and a data representation file, that performs a process based on the values of the parameters in the input and output hashtables,

wherein the data of the first hashtable and arguments of the second hashtable are consistent with one or more data types specified within the data representation file.

32. (Previously Presented) A memory for storing data associated with a custom activity for access by a process being executed by a processor, the memory comprising:

a structure specifying an input tag that obtains a value for an input hashtable to be used as an argument in a method, specifying an output tag that specify parameters that define what to do with parameters in an output hashtable including output arguments associated with the method, and specifying design tags that define a user interface associated with the custom activity associated with a class file and a data representation file,

wherein the data of the first hashtable and arguments of the second hashtable are consistent with one or more data types specified within the data representation file.

33. (Original) The structure of claim 32, wherein the input, output and design tags each include parameter tags that have attributes defining user interface characteristics associated with the respective tag.

34. (Previously Presented) A memory for storing data for access by a process being executed by a processor, the memory comprising:

a structure defining a custom activity implemented in a process management system by defining a package for importing packages external to the process management system, defining an init() method for defining initialization tasks associated with the custom activity, and defining a perform() method for executing tasks associated with the custom activity,

wherein the perform() method is associated with at least a first hashtable including values corresponding to data fields and a second hashtable including values to be placed in the data fields.

35. (Canceled)

36. (Original) The structure of claim 34, wherein the custom activity may have a plurality of instances and wherein the init() method defines an association with resources external to the process management system and are shared by all instances of the custom activity.

37. (Withdrawn) A system for creating and implementing custom activities in a process management environment, comprising:

a processor; and

a memory containing instructions executable by the processor to:

receive a selection to add a custom palette;

receive information reflecting an identifier associated with the custom palette; and

assigning a visual representation to the custom palette reflecting a custom activity that may be used in an automated workflow process to access information external to the process management environment,

wherein at least two applications employ the custom activity, each class of the at least two applications is deployed to an appropriate folder in a class path such that the class path is shared by the at least two applications, and subsequent deployments of the at least two applications overwrite the previous application class that was deployed to the class path.

38. (Withdrawn) A system for creating and implementing a custom activity in a process managements environment, comprising:

a processor; and

a memory containing instructions executable by the processor to:

receive a request to generate a palette associated with the custom activity; assign the custom activity to the palette; and

determine activation of the custom activity based on a manipulation associated with the palette, wherein the custom activity accesses resources external to the process management environment,

wherein at least two applications employ the custom activity, each class of the at least two applications is deployed to an appropriate folder in a class path such that the class path is shared by the at least two applications, and subsequent deployments of the at least two applications overwrite the previous application class that was deployed to the class path.

39. (Currently Amended) A system for creating and implementing a custom activity in a process managements environment, comprising:

a processor; and

a memory containing instructions executable by the processor to:

receive a first file reflecting a class file that implements an interface and defining with an interface with a package external to the process management system;

receive a second file defining parameters that the first file uses;

archive the first and second file in an archive file; and

invoke the first and second file based on a manipulation of an image reflecting the custom activity in a visual process map reflecting an automated workflow process,

wherein the first file and second file have the same root name and, once the archive file is created, a directory structure is checked to ensure that the directory structure reflects the package structure of class file.

40. (Previously Presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for creating an activity within a process management system, the method comprising:

receiving first data reflecting a class file that implements an interface;

receiving second data reflecting a data representation file that specifies the environment, input parameters, and output parameters for the class;

packaging the first and second data; and associating the packaged data with an activity that may be used in an automated workflow process to access information external to the process management system.

41. (Original) The computer-readable medium of claim 40, wherein the data representation file includes a section that determines the appearance of a representation reflecting the activity.

42. (Original) The computer-readable medium of claim 40, wherein the class file includes a method that is configured to obtain a value of a parameter defined in the data representation file.

43. (Original) The computer-readable medium of claim 40, wherein receiving first data reflecting a class file includes:

receiving data that defines a package for the class file; and

receiving data that defines methods that retrieve and set values to variables to be used by the activity.

44. (Original) The computer-readable medium of claim 43, wherein receiving data that defines methods includes:

receiving data that reflects a method that defines variables that are constant across all instances of the activity.

45. (Original) The computer-readable medium of claim 44, wherein the method is associated with an input hashtable to define values of a variable used by the activity

46. (Original) The computer-readable medium of claim 43, wherein receiving data that defines methods includes:

receiving data reflecting a method that defines values for variables in a first hashtable and retrieves values for variables from a second hashtable.

47. (Original) The computer-readable medium of claim 43, wherein receiving data that defines methods includes:

receiving data reflecting a method that releases resources used by an application that implements the activity when the application is unloaded from the process management system.

48. (Original) The computer-readable medium of claim 40, wherein receiving second data reflecting a data representation file includes:

receiving data reflecting a first section that defines a type and name of the class file;

receiving data reflecting a second section that defines parameters with values that remain constant within all instances of the activity;

receiving data reflecting a third section that sets values for selected parameters within a first hashtable;

receiving data reflecting a fourth section that defines what to do with parameters included in a second hashtable; and

receiving data reflecting a fifth section associated with a visual representation associated with the activity.

49. (Original) The computer-readable medium of claim 40, wherein packaging the first and second data includes:

packaging the first and second data into one of a JAR file or a ZIP file.

50. (Original) The computer-readable medium of claim 40, wherein associating the packaged data with an activity includes:

locating the packaged data; and

receiving data reflecting a visual representation that corresponds to the packaged files.

51. (Previously Presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for implementing a custom activity within a process management environment, the method comprising:

defining a file associated with a custom activity;

assigning a visual representation associated with the custom activity;

receiving an indication reflecting implementation of the custom activity in a workflow process based on a position of the visual representation in a process map representing the workflow process; and

invoking the file,

wherein the file includes a class file that implements an interface and a data representation file that specifies the environment, input parameters, and output parameters for the class.

52. (Original) The computer-readable medium of claim 51, wherein the file is an archive file an includes the visual representation.

53. (Currently Amended) A computer-readable medium including instructions for performing a method, when executed by a processor, for creating and defining a custom activity within a process management system, the method comprising:

creating at least one file defining properties associated with the custom activity; and

defining a model associated with the custom activity, wherein the custom activity may be used to access information external to the process manager system[[.]],

wherein ~~a first~~ the file includes ~~is~~ a class file that implements an interface and ~~a second file is~~ a data representation file that specifies the environment, input parameters, and output parameters for the class.

54. (Original) The computer-readable medium of claim 53, wherein the model is an image reflecting the custom activity.

55. (Original) The computer-readable medium of claim 53, further comprising:

packaging the file and model into an archive file.

56. (Original) The computer-readable medium of claim 53, further comprising:

associating the custom activity with a workflow process managed by the process management system.

57. (Original) The computer-readable medium of claim 56, wherein associating the custom activity includes:

determining a position of the model in a visual process map reflecting the workflow process; and

invoking the custom activity in the workflow process based on the determination.

58. (Original) The computer-readable medium of claim 53, wherein the at least one file includes a Java class file and an XML description file.

59. (Withdrawn) A computer-readable medium including instructions for performing a method, when executed by a processor, for implementing a custom activity in a process management system, the method comprising:

creating a process map reflecting an automated workflow process;

creating an image reflecting a custom activity; and

invoking a class defining the custom activity based on a manipulation of the image by a user such that the image is placed in the process map,

wherein the custom activity exchanges data with resources external to the process management system,

wherein at least two applications employ the custom activity, each class of the at least two applications is deployed to an appropriate folder in a class path such that the class path is shared by the at least two applications, and subsequent deployments of the at least two applications overwrite the previous application class that was deployed to the class path.

60. (Currently Amended) A computer-readable medium including instructions for performing a method, when executed by a processor, for creating a custom activity in a process management system, the custom activity exchanging information with resources external to the process management system, the method comprising:

receiving a first file reflecting a class file that implements an interface and a second file; and

archiving the files in an archive file such that when the custom activity is activated the archived files are accessed and executed,

wherein the first file and second file have the same root name and, once the archive file is created, a directory structure is checked to ensure that the directory structure reflects the package structure of class file.

61. (Original) The computer-readable medium of claim 60, wherein receiving the first and second files includes:

receiving package information associated with the first file that implement packages external to the process management system.

62. (Original) The computer-readable medium of claim 60, wherein receiving the first and second files includes:

receiving data that interacts with parameters associated with a hashtable defined in the second file.

63. (Original) The computer-readable medium of claim 60, wherein receiving the class and XML files includes:

receiving data associated with the second file that defines at least one hashtable used by the first file.

64. (Original) The computer-readable medium of claim 60, wherein the first file reflects a class file and the second file reflects an XML file.

65. (Original) The computer-readable medium of claim 60, wherein archiving the files includes:

archiving the files in an archive file consisting of one of a JAR file and a ZIP file.

66. (Previously Presented) A system for creating an activity within a process management system, comprising:

means for receiving first data reflecting a class file that implements an interface;

means for receiving second data reflecting a data representation file that specifies the environment, input parameters, and output parameters for the class;

means for packaging the first and second data; and

means for associating the packaged data with an activity that may be used in an automated workflow process to access information external to the process management system.

67. (Original) The system of claim 66, wherein the data representation file includes a section that determines the appearance of a representation reflecting the activity.

68. (Original) The system of claim 66, wherein the class file includes a method that is configured to obtain a value of a parameter defined in the data representation file.

69. (Original) The system of claim 66, wherein the means for receiving first data reflecting a class file includes:

means for receiving data that defines a package for the class file; and means for receiving data that defines methods that retrieve and set values to variables to be used by the activity.

70. (Original) The system of claim 69, wherein the means for receiving data that defines methods includes:

means for receiving data that reflects a method that defines variables that are constant across all instances of the activity.

71. (Original) The system of claim 70, wherein the method is associated with an input hashtable to define values of a variable used by the activity

72. (Original) The system of claim 69, wherein the means for receiving data that defines methods includes:

means for receiving data reflecting a method that defines values for variables in a first hashtable and retrieves values for variables from a second hashtable.

73. (Original) The system of claim 69, wherein the means for receiving data that defines methods includes:

means for receiving data reflecting a method that releases resources used by an application that implements the activity when the application is unloaded from the process management system.

74. (Original) The system of claim 66, wherein the means for receiving second data reflecting a data representation file includes:

means for receiving data reflecting a first section that defines a type and name of the class file;

means for receiving data reflecting a second section that defines parameters with values that remain constant within all instances of the activity;

means for receiving data reflecting a third section that sets values for selected parameters within a first hashtable;

means for receiving data reflecting a fourth section that defines what to do with parameters included in a second hashtable; and

means for receiving data reflecting a fifth section associated with a visual representation associated with the activity.

75. (Original) The system of claim 66, wherein the means for packaging the first and second data includes:

packaging the first and second data into one of a JAR file or a ZIP file.

76. (Original) The system of claim 66, wherein the means for associating the packaged data with an activity includes:

means for locating the packaged data; and means for receiving data reflecting a visual representation that corresponds to the packaged files.

77. (Previously Presented) A system for implementing a custom activity within a process management environment, comprising:

means for defining a file associated with a custom activity;

means for assigning a visual representation associated with the custom activity; means for receiving an indication reflecting implementation of the custom activity in a workflow process based on a position of the visual representation in a process map representing the workflow process; and

means for invoking the file,

wherein the file includes a class file that implements an interface and a data representation file that specifies the environment, input parameters, and output parameters for the class.

78. (Original) The system of claim 77, wherein the file is an archive file an includes the visual representation.

79. (Currently Amended) A system for creating and defining a custom activity within a process management system, comprising:

means for creating at least one file defining properties associated with the custom activity; and

means for defining a model associated with the custom activity, wherein the custom activity may be used to access information external to the process manager system,

wherein ~~a first~~ the file includes ~~is~~ a class file that implements an interface and ~~a second file is~~ a data representation file that specifies the environment, input parameters, and output parameters for the class.

80. (Original) The system of claim 79, wherein the model is an image reflecting the custom activity.

81. (Original) The system of claim 79, further comprising:

means for packaging the file and model into an archive file.

82. (Original) The system of claim 79, further comprising:

means for associating the custom activity with a workflow process managed by the process management system.

83. (Original) The system of claim 82, wherein the means for associating the custom activity includes:

means for determining a position of the model in a visual process map reflecting the workflow process; and

means for invoking the custom activity in the workflow process based on the determination.

84. (Original) The system of claim 79, wherein the at least one file includes a Java class file and an XML description file.

85. (Withdrawn) A system for implementing a custom activity in a process management system, comprising:

means for creating a process map reflecting an automated workflow process;

means for creating an image reflecting a custom activity; and

means for invoking a class defining the custom activity based on a manipulation of the image by a user such that the image is placed in the process map, wherein the

custom activity exchanges data with resources external to the process management system,

wherein at least two applications employ the custom activity, each class of the at least two applications is deployed to an appropriate folder in a class path such that the class path is shared by the at least two applications, and subsequent deployments of the at least two applications overwrite the previous application class that was deployed to the class path.

86. (Currently Amended) A system for creating a custom activity in a process management system, the custom activity exchanging information with resources external to the process management system, comprising:

means for receiving a first file reflecting a class file that implements an interface and a second file; and

means for archiving the files in an archive file such that when the custom activity is activated the archived files are accessed and executed,

wherein the first file and second file have the same root name and, once the archive file is created, a directory structure is checked to ensure that the directory structure reflects the package structure of class file.

87. (Original) The system of claim 86, wherein the means for receiving the first and second files includes:

means for receiving package information associated with the first file that implement packages external to the process management system.

88. (Original) The system of claim 86, wherein the means for receiving the first and second files includes:

means for receiving data that interacts with parameters associated with a hashtable defined in the second file.

89. (Original) The system of claim 86, wherein the means for receiving the class and XML files includes:

means for receiving data associated with the second file that defines at least one hashtable used by the first file.

90. (Original) The system of claim 86, wherein the first file reflects a class file and the second file reflects an XML file.

91. (Original) The system of claim 86, wherein the means for archiving the files includes:

means for archiving the files in an archive file consisting of one of a JAR file and a ZIP file.